(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 June 2002 (27.06.2002)

PCT

(10) International Publication Number
WO 02/50670 A1

(51) International Patent Classification[7]: G06F 9/44

(21) International Application Number: PCT/NL01/00926

(22) International Filing Date:
19 December 2001 (19.12.2001)

(25) Filing Language: Dutch

(26) Publication Language: English

(30) Priority Data:
1016958 21 December 2000 (21.12.2000) NL

(71) Applicant *(for all designated States except US)*: **CROSSMARX B.V.** [NL/NL]; P.O. Box 11172, NL-1001 GD Amsterdam (NL).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **VOGLER, Michel** [NL/NL]; Nieuwe Achtergracht 73, NL-1018 WL Amsterdam (NL).

(74) Agent: **JORRITSMA, Ruurd**; Nederlandsch Octrooibureau, Scheveningseweg 82, P.O. Box 29720, NL-2502 LS The Hague (NL).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

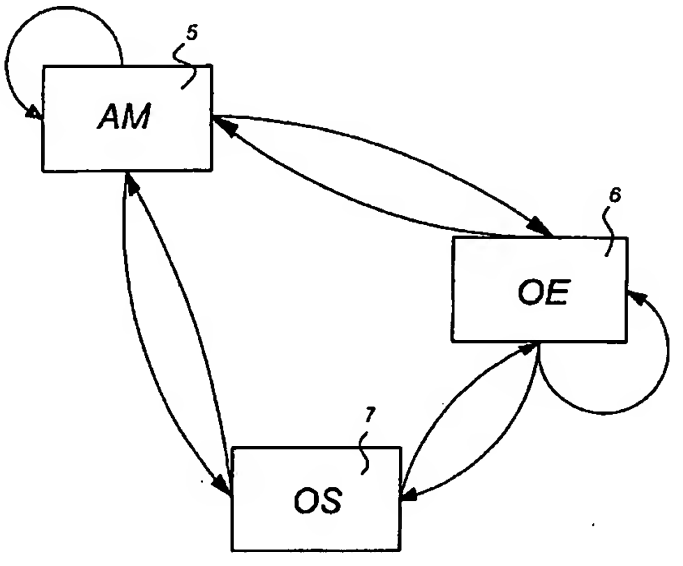
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

[Continued on next page]

(54) **Title:** SOFTWARE APPLICATION AND METHOD FOR BUILDING SOFTWARE APPLICATIONS



(57) **Abstract:** Method for building software applications, comprising the steps of linking a plurality of modules (5, 6, 7), each of which contain a piece of program code, to form a graph, defining the configuration of the software application by defining at least one object model on the basis of at least one entity comprising an entity identifier and an entity name, and at least on attribute associated with the entity, comprising an attribute identifier, an attribute name, a data type and an identifier for the entity to which the attribute points, wherein each module (5, 6, 7) is arranged to communicate, during operation, a context to a following module (5, 6, 7) that is linked to the module (5, 6, 7) in the graph, the context determining the behaviour of the following module. The invention also relates to a software application that is produced using the method.

WO 02/50670 A1

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Software application and method for building software applications

The present invention relates to a software application and a method for building software applications.

At present software applications are produced in various ways. In a first form a software application is designed as a custom application. This has the disadvantages that it is expensive and time-consuming, can contain many errors and has high maintenance costs. In addition, the company using the application runs the risk that the employee with knowledge of the application leaves the company taking all his/her knowledge with him/her.

In a second form use is made of standard packages in order to obtain a software application with a certain functionality. Disadvantages of such an approach are that the application has little flexibility, will have a large amount of unnecessary functionality and is difficult to integrate with other software packages and the fact that a high degree of dependence on the supplier results.

The aim of the present invention is to provide a software application and a method for building software applications making use of standard modules only, where the disadvantages of the known methods do not arise or arise to an appreciably lesser extent.

Said aim is achieved by a software application of the type defined in the preamble, wherein the software application comprises a plurality of modules, wherein each module comprises a piece of program code, wherein the plurality of modules comprises mutual links between modules in order to form a graph, wherein each of the plurality of modules contains configuration data for defining at least one object model on the basis of at least one entity comprising an entity identifier and an entity name, and at least one attribute associated with the entity, comprising an attribute identifier, an attribute name, a data type and an identifier for the entity to which the attribute refers, and wherein during operation a context is communicated by a first module to a following module that is mutually linked to the first module in the graph, wherein the context determines the behaviour of the following module.

Such a software application has the advantage that technically it requires no customising. Effort is required only for the definition of the functionality of the software application by linking the modules to produce a graph and the definition of the configu-

ration of the modules. Consequently it will be possible to produce a software application much more quickly and the risk of programming errors is appreciably reduced.

In a preferred embodiment of the software application according to the present invention, the plurality of modules is formed by a plurality of standard modules and each standard module contains program code with a predetermined base functionality and a specific functionality determined by the configuration data.

By making use of standard modules with a certain basic functionality (such as display of data) and a specific functionality (such as display of data in a list of specific dimensions and with specific layout characteristics) determined by the configuration data it is possible to build an extensive software application within a short time. Errors will be prevented as far as possible by the standard basic functionality and consistent use of the configuration data in the standard modules.

In one embodiment the data type of the at least one attribute is a connection, in order to indicate that the at least one attribute is a reference to a further entity, the at least one attribute also containing the identifier for said further entity. This makes it possible to communicate the context from a calling module to a called module, the calling module being able to influence the behaviour of the called module.

In a further embodiment of the software application according to the present invention the context comprises an attribute restrictor, comprising a tuple of an attribute identifier and an attribute value and/or an entity restrictor, comprising a tuple of an entity identifier and an attribute value. Such measures make it possible for a calling module to influence the behaviour of a called module by imposing restrictions on the called module.

The software application according to the present invention can comprise a number of standard modules which each fulfil a specific function. Preferably, the plurality of modules contains an action menu block for presenting choices and giving commands.. In addition, the plurality of modules can contain an object search block for searching for data. The plurality of modules can also contain an object editing block for presenting data on a screen and editing the presented data. A user interface for a software application can be compiled with the aid of these standard modules.

A second aspect of the present invention relates to a method for building software applications, comprising the steps of linking a plurality of modules, each of which contain a piece of program code, to form a graph, defining the configuration of the

software application by defining at least one object model on the basis of at least one entity comprising an entity identifier and an entity name, and at least one attribute associated with the entity, comprising an attribute identifier, an attribute name, a data type and an identifier for the entity to which the attribute refers, wherein each module is arranged to communicate, during operation, a context to a following module that is linked to the module in the graph, the context determining the behaviour of the following module.

Further embodiments of the present method, corresponding to the abovementioned embodiments of the software application according to the present invention, are defined in appended Claims 10 to 16.

Using the present method it is possible to build software applications in a simple and rapid manner, the advantages cited above in the description of the software application also being obtained.

In a further aspect of the present invention, a software program product is provided, such as a compact disc, a floppy disc or a website on a server, containing a software application according to one of Claims 1 to 8.

In yet a further aspect of the present invention a software program product is provided that contains program code which, after loading, provides a processing system with the functionality of the method according to one of Claims 9 to 16.

The present invention will be explained in more detail below on the basis of an example, with reference to the appended drawings, in which:

Fig. 1 shows an overview of a number of standard modules for a software application according to the present invention and the possible links between the standard modules;

Fig. 2 shows an example of a graph of linked standard modules which forms a software application according to the present invention;

Figs. 3a to 3d show examples of screens that can be displayed by the software application example.

A number of standard type modules which in a software application make interaction with a user possible are shown in Fig. 1. Fig. 1 further shows which links between the modules are possible. The first standard module shown is an action menu 5. The action menu 5 is a module that displays buttons by means of which the user can call another module. The second standard module shown is an object editor 6, by

means of which data can be manipulated. The final module shown is an object searcher 7, by means of which it is possible to search for data.

The standard modules 5, 6 and 7 shown in Fig. 1 can be linked to one another in two directions. The action menu module 5 and the object editor module 6 can also be linked to a module of the same type. It is not possible to call up a further object search module 7 from the object search module 7 since it can never be functional to call up another search screen from one search screen within a software application. New standard modules 5, 6, 7 are added to an application by linking to an already existing module, indicated by the arrows in Fig. 1. It can be deduced from Fig. 1 that (with the use of three standard modules 5, 6, 7) there are 8 types of links in total, all of which have a different functional effect within the software application.

During operation or running of the software application a specific module 5, 6, 7 calls the following module 5, 6, 7 in accordance with the graph made. The context of each module 5, 6, 7 is determined by the status of relevant surrounding modules 5, 6, 7. By making use of calls, the context of surrounding modules 5, 6, 7 is transmitted to the called module 5, 6, 7. This can take place directly or indirectly via a global data structure.

The configuration of a software application according to the present invention comprises a number of details, inter alia a description of an object model, which, in turn, is described on the basis of entities and attributes. An entity comprises at least an entity identifier (a unique number) and an entity name. An attribute comprises at least an attribute identifier (a unique number), a name, a data type and an indicator for the entity to which the attribute part refers. The data type can be, for example, an integer or a string. The data type can, however, also be of the 'connection' type, which indicates that an attribute with such a data type is a reference to a further entity. In this case the identifier of the further entity is also incorporated in the attribute. The attribute can also form part of the key for the entity to which it belongs, as a result of which the entity can be uniquely identified with the aid of this attribute.

According to the invention a software application is able to function in that modules 5, 6, 7 call one another in accordance with a graph and in doing so transmit a context to a following module 5, 6, 7. This context determines the behaviour of the module. The context that is transmitted by a calling module 5, 6, 7 to a called module 5, 6, 7 comprises a list containing attribute restrictors and/or entity restrictors. An attribute

restrictor is a tuple of an attribute identifier and an attribute value. For this purpose the entities which the called module 5, 6, 7 is going to process and the attributes which belong to these must be known in the calling module 5, 6, 7. In some cases the calling module 5, 6, 7 does not know which entities the called module 5, 6, 7 is going to process. In this case an entity restrictor (a tuple of an entity identifier and an attribute value) can be transmitted as context.

At the start of a software application the context is empty. When a second module 5, 6, 7 is called by a first module 5, 6, 7 an attribute restrictor and/or an entity restrictor can be added to the context. This is dependent on the type of module 5, 6, 7 and the location within the module 5, 6, 7 where the call takes place (as determined by the configuration of the module 5, 6, 7 concerned).

If an attribute restrictor is added, a restriction will be made when the second module 5, 6, 7 initialises, where the name of the attribute having an identifier that is identical to the attribute identifier of the attribute restrictor is the same as the attribute value of the attribute restrictor.

If an entity restrictor is added, the entity identifier is identical to the identifier of the entity that is addressed in the first module 5, 6, 7. When the second module 5, 6, 7 initialises a search will be made, per entity restrictor in the context, for an attribute with an entity identical to the entity(ies) that is (are) processed in the second module 5, 6, 7 and with a connection entity that is identical to the entity from the entity restrictor. If an attribute is found that meets these criteria, the identifier of that attribute is returned as the result. Together with the attribute value from the entity restrictor, the restriction that the name of the attribute with the identifier found is the same as the attribute value of the entity restrictor is applied.

The list of attribute restrictors and entity restrictors (the context) thus provides a number of restrictions which apply to all objects in the second module 5, 6, 7.

Fig. 2 shows a graph of an example of a software application according to the present invention. As the start of the graph, the software application has an action menu module 5a, that is linked to a first object search module 7a and a second object search module 7b. The first object search module 7a is connected by a two-way link to a first object editor module 6a, so that the first object search module 7a is able to call the first object editor module 6a, but also vice versa. In the same way the second object search module 7b is connected by a two-way link to a second object editor module 6b. The

graph also further comprises a two-way link between the first object editor module 6a and the second object editor module 6b.

In the case of the modules and the possible links shown in Fig. 1, an attribute restrictor or entity restrictor has to be added to the context only in certain cases. In the case of the link from an object editor module 6 to a further object editor module 6 an entity restrictor is added if a selected data element in a first object editor module 6 has to be displayed in a second linked object editor module 6 or if related data for a selected data element in a first object editor module 6 have to be displayed in a second linked object editor module 6. In the case of the link of an object editor module 6 to an object search module 7 an entity restrictor is added to the context if a search has to be made for data related to a selected data element. In the case of a reverse link (from an object search module 7 to an object editor module 6) an attribute restrictor is added to the context if a selected data element in the object search module 7 has to be displayed in an object editor module 6.

The example shown in Fig. 2 relates to an application with regard to a company that consists of departments and employees. According to a data model there are two entities, Department and Employee, and several staff belong to one Department. In the example the Department entity has two attributes, a department identifier (DeptID) and department name (DeptName) and the Employee entity has three attributes, an employee identifier (EmpID), an employee name (EmpName) and a Department. The identifier attributes are the keys for the entities. The Department attribute points to the Department entity. According to the example, the entities contain the data as shown in the following table:

Department		Employee		
DeptID	DeptName	EmpID	EmpName	Department
101	Board	201	Jansen	101
102	Planning	202	Klaasen	102
103	Production	203	Karelse	101
		204	Pietersen	103
		205	Kok	103
		206	Hulshof	102

Thus, there are three departments, the Board where Jansen and Karelse work, Planning where Klaasen and Hulshof work and finally Production where Pietersen and Kok work.

The first object search module 7a, by means of which a search for departments can be carried out, can be called up from the action menu module 5a. From the first object search module 7a it is possible to call up the first object editor module 6a to view or edit the data for a department found. It is then possible to return to the first object search module 7a for a new search action or to call up the second object editor module 6b in order to view or edit the data for the associated employees. Analogously, it is also possible to search for employees from the action menu module 5a and to view data for an employee.

The configuration of this application comprises at least the following entities and attributes:

Entities			Attributes					
ID	Name	...	ID	Entity	Name	Data type	Connection entity	...
11	Department		21	11	DeptID	Integer		
12	Employee		22	11	DeptName	String		
			23	12	EmpID	Integer		
			24	12	EmpName	String		
			25	12	Department	Connection	11	

If the software application is run it starts at the action menu module 5a, which, for example, ensures that a screen as shown in Fig. 3a is displayed. Two buttons 21, 22, which are labelled Department and Employee respectively, are defined in the configuration of the action menu module 5a, which buttons each start the (relevant) associated object search module 7a, 7b on clicking thereon. The context is empty and thus has no influence on the behaviour of the action menu module 5a. If the user clicks on button 21 (Department), the first object search module 7a is then started. In the configuration of this module it is defined that a search will be made for data on entity identifier 11. The context of the first object search module 7a is empty; after all, nothing is added to the context when the first object search module 7a is called up by the action menu module 5a. This means that no restrictions are imposed during the search for depart-

ments. If, for example, the search is for all departments, all departments will be displayed in a table on the screen by the first object search module 7a, as shown in Fig. 3b. The screen then shows a table with title 23 (Name) and three rows 24, 25, 26 containing the names of the existing Departments (Board, Planning and Production respectively).

If the Planning department (row 25) is selected in the screen displayed by the first object search module 7a, the first object editor module 6a is started. During this operation an attribute restrictor, that is to say the tuple [21,102] is added to the context, where 21 is the identifier of the attribute and 102 is the identifier of the department. In the configuration of the first object editor module 6a to be called up it is now defined that data for entity 11 must be displayed. If the context were to be empty, all departments would then be displayed in this first object editor module 6a. Because the first object search module 7a has added the attribute restrictor [21,102] to the context, the restriction [DeptID = 102] is applied when retrieving the departments, as a result of which only the Planning department is displayed, as shown in Fig. 3c, with a second screen 27, in which two fields are displayed, that is to say a first field 28 (102) and a second field 29 (Planning).

The employees in the Planning department can then be displayed in the second object editor module 6b by clicking on a button in the screen displayed by the first object editor module 6a. For this purpose the first object editor module 6a adds an entity restrictor to the context, specifically the tuple [11,102], where 11 is the identity of the entity and 102 the identity of the department. In the configuration of the second object editor module 6b it is now defined that data for entity 12 have to be displayed. Before the second object editor module 6b starts to load data, a search will now first be made for an attribute of an entity with identifier 12 with entity 11 as connection. The attribute with attribute identifier 25 meets these criteria. The second object editor module 6b now applies Department = 102 as additional restriction when retrieving the employees. As a consequence of this only the employees of the Planning department are displayed in the screen displayed by the second object editor module 6b (see Fig. 3d). In addition to the second screen 27 shown in Fig. 3c, a third screen 30 is now also displayed, containing two fields 31, 32, containing the names of the two employees of the Planning department (Klaasen and Hulshof, respectively). The other links in the application example function in the same way as the links already described and are therefore not

further described in more detail.

In the example shown the context determines how a specific object (such as the display screen 30 in Fig. 3d) behaves (or the appearance of this object). As will be clear to those skilled in the art, the context can be used to determine the behaviour or the appearance of multiple objects. In this context consideration can be given, for example, to the display in a separate list of additional data relating to the employees displayed in fields 31 and 32, which additional data relate solely to the department-related data for the employees.

CLAIMS

1. Software application comprising a plurality of modules (5, 6, 7),

wherein each module (5, 6, 7) comprises a piece of program code,

wherein the plurality of modules (5, 6, 7) comprises mutual links between modules in order to form a graph,

wherein each of the plurality of modules (5, 6, 7) contains configuration data for defining at least one object model on the basis of at least one entity comprising an entity identifier and an entity name, and at least one attribute associated with the entity, comprising an attribute identifier, an attribute name, a data type and an identifier for the entity to which the attribute points,

and wherein during operation a context is communicated by a first module (5, 6, 7) to a following module (5, 6, 7) that is linked to the first module (5, 6, 7) in the graph, wherein the context determines the behaviour of the following module (5, 6, 7).

2. Software application according to Claim 1, wherein the plurality of modules (5, 6, 7) is formed by a plurality of standard modules and each standard module (5, 6, 7) contains program code with a predetermined base functionality and a specific functionality determined by the configuration data.

3. Software application according to Claim 1 or 2, wherein the data type of the at least one attribute is a connection, in order to indicate that the at least one attribute is a reference to a further entity, the at least one attribute also containing the identifier for said further entity.

4. Software application according to Claim 1, 2 or 3, wherein the context contains an attribute restrictor, comprising a tuple of an attribute identifier and an attribute value.

5. Software application according to one of the preceding claims, wherein the context contains an entity restrictor, comprising a tuple of an entity identifier and an attribute value.

6. Software application according to one of the preceding claims, wherein the plurality of modules (5, 6, 7) contains an action menu block (5) for presenting choices and giving commands.

7. Software application according to one of the preceding claims, wherein the plurality of modules (5, 6, 7) contains an object search block (7) for searching for data.

8. Software application according to one of the preceding claims, wherein the plurality of modules (5, 6, 7) contains an object editing block (6) for presenting data on a screen and editing the presented data.

9. Method for building software applications, comprising the steps of:

linking a plurality of modules (5, 6, 7), each of which contain a piece of program code, to form a graph;

defining the configuration of the software application by defining at least one object model on the basis of at least one entity comprising an entity identifier and an entity name, and at least one attribute associated with the entity, comprising an attribute identifier, an attribute name, a data type and an identifier for the entity to which the attribute refers,

wherein each module (5, 6, 7) is arranged to communicate, during operation, a context to a following module (5, 6, 7) that is linked to the module (5, 6, 7) in the graph, the context determining the behaviour of the following module.

10. Method according to Claim 9, wherein the plurality of modules (5, 6, 7) is formed by a plurality of standard modules and each standard module (5, 6, 7) contains program code with a predetermined base functionality and a specific functionality determined by the configuration data.

11. Method according to Claim 9 or 10, wherein the data type of the at least one attribute is a connection, in order to indicate that the at least one attribute is a reference to a further entity, the at least one attribute also containing the identifier for said further entity.

12. Method according to Claim 9, 10 or 11, wherein the context contains an attribute restrictor, comprising a tuple of an attribute identifier and an attribute value.

13. Method according to one of Claims 9 to 12, wherein the context contains an entity restrictor, comprising a tuple of an entity identifier and an attribute value.

14. Method according to one of Claims 9 to 13, wherein the plurality of modules (5, 6, 7) contains an action menu block (5) for presenting choices and giving commands.

15. Method according to one of Claims 9 to 14, wherein the plurality of modules (5, 6, 7) contains an object search block (7) for searching for data.

16. Method according to one of Claims 9 to 15, wherein the plurality of modules (5, 6, 7) contains an object editing block (6) for presenting data on a screen and editing the presented data.

17. Software program product comprising a software application according to one of Claims 1 to 8.

18. Software program product comprising program code which, after loading, provides a processing system with the functionality of the method according to one of Claims 9 to 16.

****

Fig 1

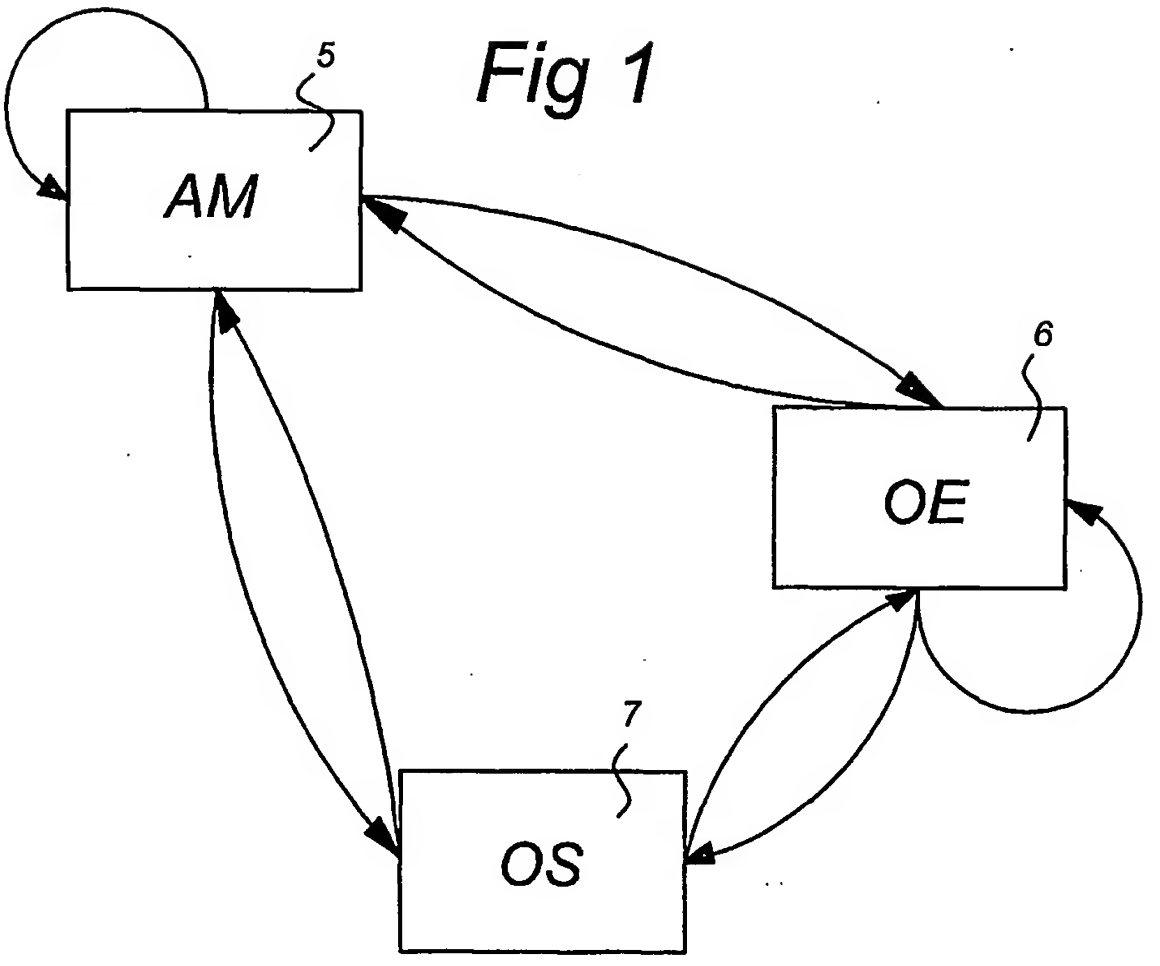


Fig 2

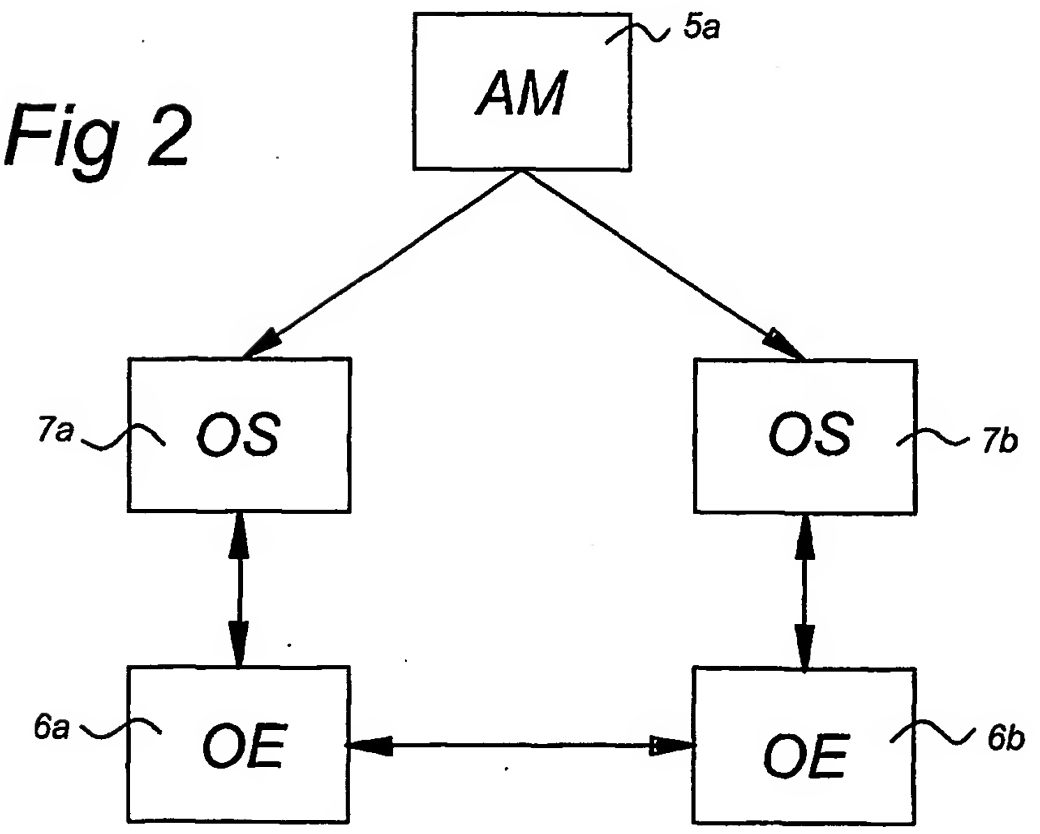


Fig 3a

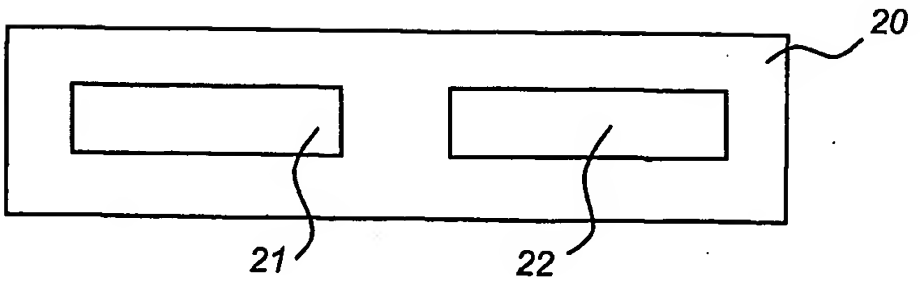


Fig 3b

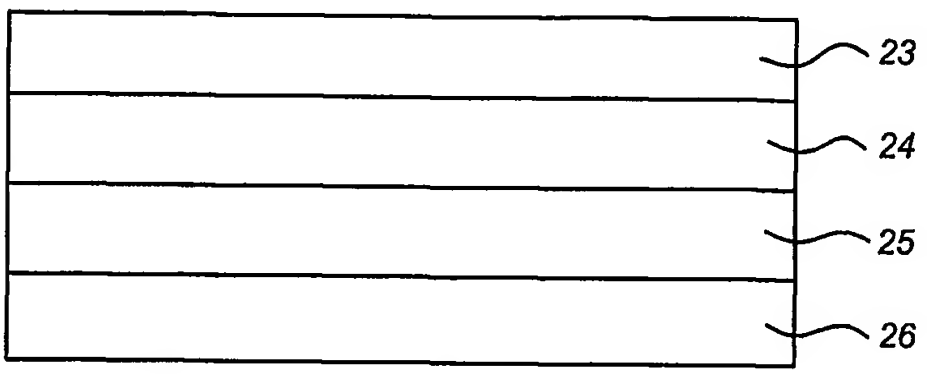


Fig 3c

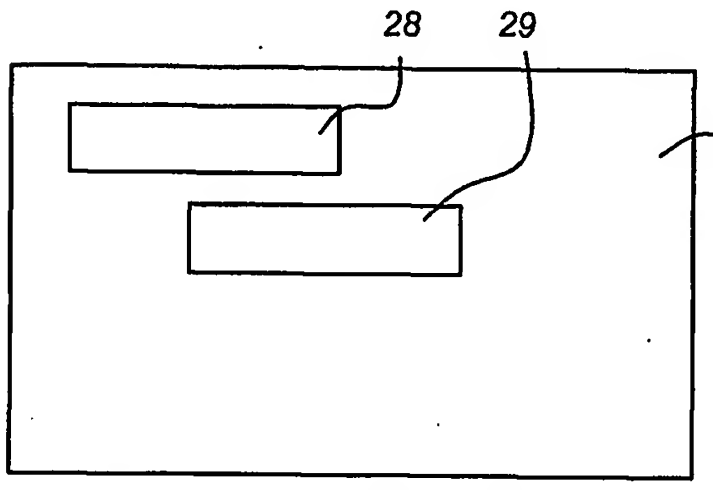
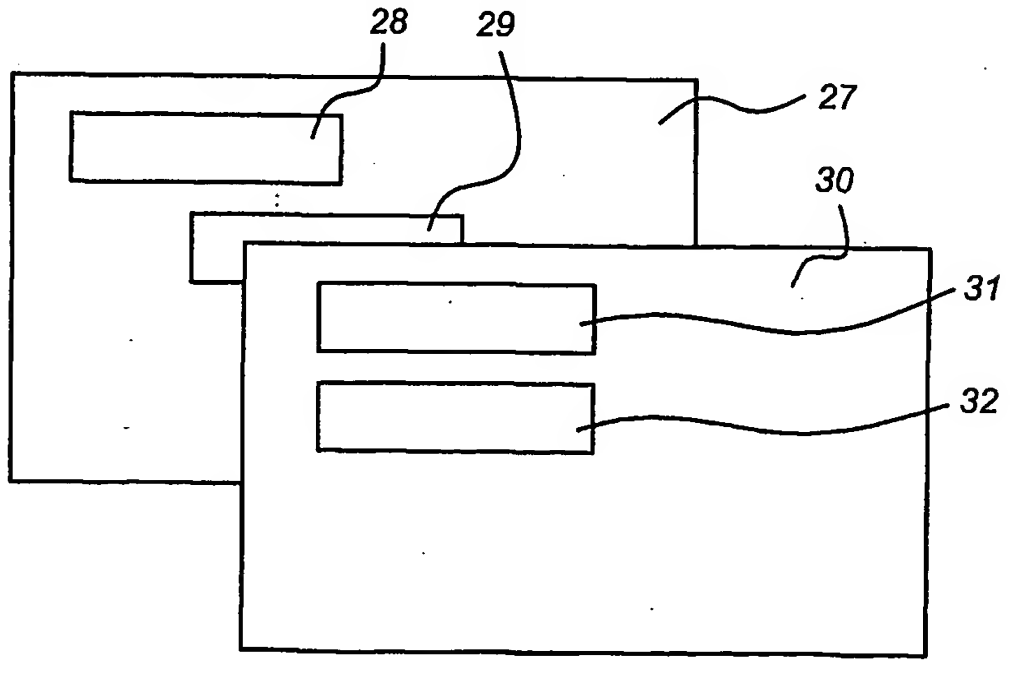


Fig 3d



INTERNATIONAL SEARCH REPORT

PCT/NL 01/00926

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, INSPEC, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 574 918 A (HURLEY DANIEL F ET AL) 12 November 1996 (1996-11-12) column 1, line 36 -column 2, line 21	1-18
A	EP 0 780 763 A (IBM) 25 June 1997 (1997-06-25) column 3, line 35 -column 5, line 19	1-18
A	US 5 381 548 A (MATSUO AKIHIKO) 10 January 1995 (1995-01-10) column 1, line 37 - line 59	1-18

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

'A' document defining the general state of the art which is not considered to be of particular relevance

'E' earlier document but published on or after the international filing date

'L' document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

'O' document referring to an oral disclosure, use, exhibition or other means

'P' document published prior to the international filing date but later than the priority date claimed

'T' later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

'X' document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

'Y' document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

'&' document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 February 2002	06/03/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Brandt, J

INTERNATIONAL SEARCH REPORT

PCT/NL 01/00926

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5574918 A	12-11-1996	US 5933637 A	03-08-1999
		CA 2128387 A1	24-02-1995
		CN 1124379 A	12-06-1996
		DE 69423158 D1	06-04-2000
		DE 69423158 T2	21-09-2000
		EP 0640914 A2	01-03-1995
		ES 2142912 T3	01-05-2000
		JP 7168710 A	04-07-1995
		US 5524246 A	04-06-1996
EP 0780763 A	25-06-1997	CA 2165893 A1	22-06-1997
		EP 0780763 A1	25-06-1997
		US 5915113 A	22-06-1999
US 5381548 A	10-01-1995	JP 5233239 A	10-09-1993